Tracing Go with eBPF

Florian Lehner

June 2021 Berlin Gophers Meetup

What is eBPF?

classic BPF

- two 32 bit registers
- SO_ATTACH_FILTER

```
(000) ldh      [12]
(001) jeq      #0x86dd        jt 2      jf 8
(002) ldb      [20]
(003) jeq      #0x6           jt 4 jf 19
(004) ldh      [54]
(005) jeq      #0x16          jt 18     jf 6
(006) ldh      [56]
(007) jeq      #0x16          jt 18     jf 19
(008) jeq      #0x800         jt 9      jf 19
(009) ldb      [23]
(010) jeq      #0x6           jt 11     jf 19
(011) ldh      [20]
(012) jset     #0x1fff        jt 19     jf 13
(013) ldxb     4*([14]&0xf)
(014) ldh      [x + 14]
(015) jeq      #0x16          jt 18     jf 16
(016) ldh      [x + 16]
(017) jeq      #0x16          jt 18     jf 19
(018) ret      #262144
(019) ret      #0
```

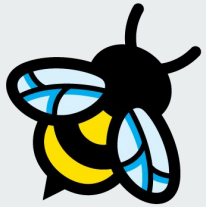
What is eBPF?

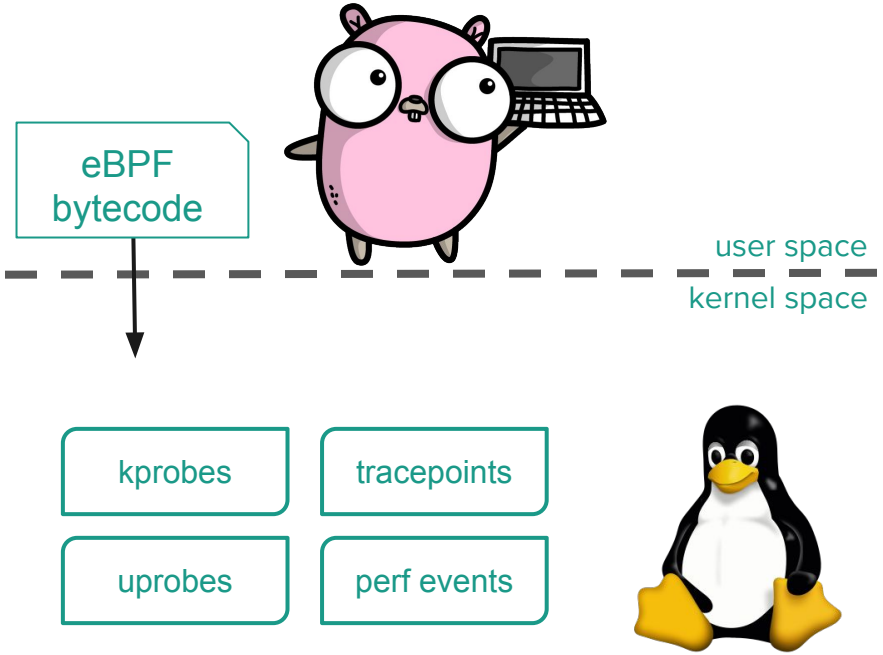
"What Javascript is to HTML, BPF is to the Linux kernel"

– Beatriz Martínez Rubio (IBM) @ KubeCon 2019

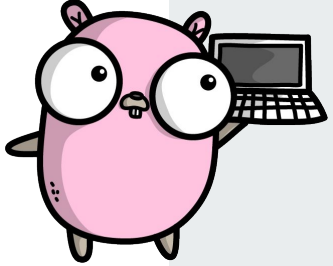
"crazy stuff"

– Alexei Starovoitov (eBPF lead)



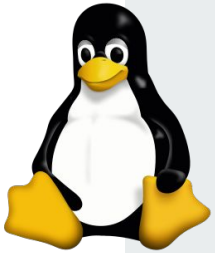


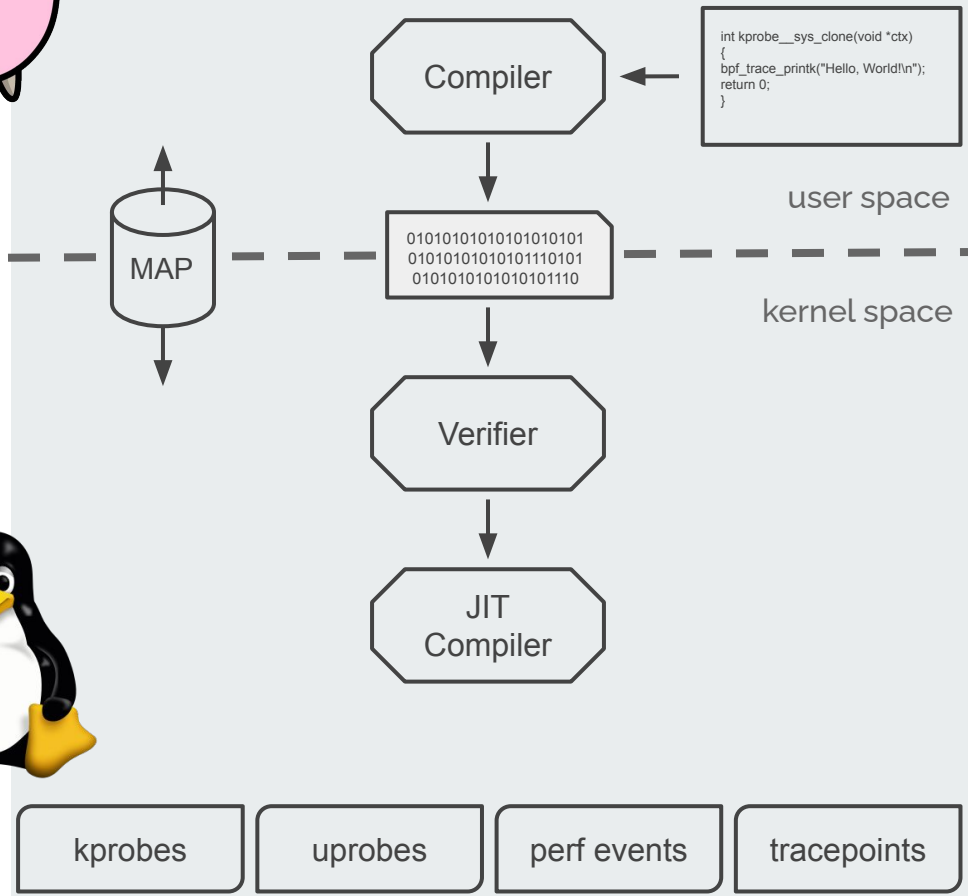
http://man7.org/linux/man-pages/man2/bpf.2.html
http://www.brendangregg.com/ebpf.html



eBPF

- eleven 64 bit registers
- 512 byte stack
- kernel side helper functions
- maximum of 1 million instruction
- maps for data exchange
- man 2 bpf





Comparison

eBPF

- eleven 64 bit registers
- 512 byte stack
- kernel side helper functions
- maximum of 1 million instruction per program
- maps for data exchange
- man 2 bpf

classic BPF

- two 32 bit registers
- SO_ATTACH_FILTER

Where do I find BPF?

```
struct bpf_insn insn[] = {
  BPF_JMP_IMM(BPF_JNE, BPF_REG_7, htobe16(protocol), 0),

  BPF_MOV64_REG(BPF_REG_1, BPF_REG_6),
  BPF_MOV32_IMM(BPF_REG_2, addr_offset),

  BPF_MOV64_REG(BPF_REG_3, BPF_REG_10),
  BPF_ALU64_IMM(BPF_ADD, BPF_REG_3, -addr_size),

  BPF_MOV32_IMM(BPF_REG_4, addr_size),
  BPF_RAW_INSN(BPF_JMP | BPF_CALL, 0, 0, 0, BPF_FUNC_skb_load_bytes),

  BPF_LD_MAP_FD(BPF_REG_1, map_fd),
  BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),
  BPF_ALU64_IMM(BPF_ADD, BPF_REG_2, -addr_size - sizeof(uint32_t)),
  BPF_ST_MEM(BPF_W, BPF_REG_2, 0, addr_size * 8),

  BPF_RAW_INSN(BPF_JMP | BPF_CALL, 0, 0, 0,
BPF_FUNC_map_lookup_elem),
  BPF_JMP_IMM(BPF_JEQ, BPF_REG_0, 0, 1),
  BPF_ALU32_IMM(BPF_OR, BPF_REG_8, verdict),
};
```

systemd

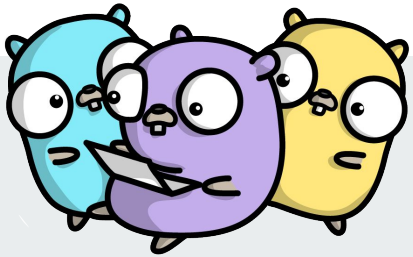
cilium

sysdig

kubernetes

https://github.com/microsoft/ebpf-for-windows

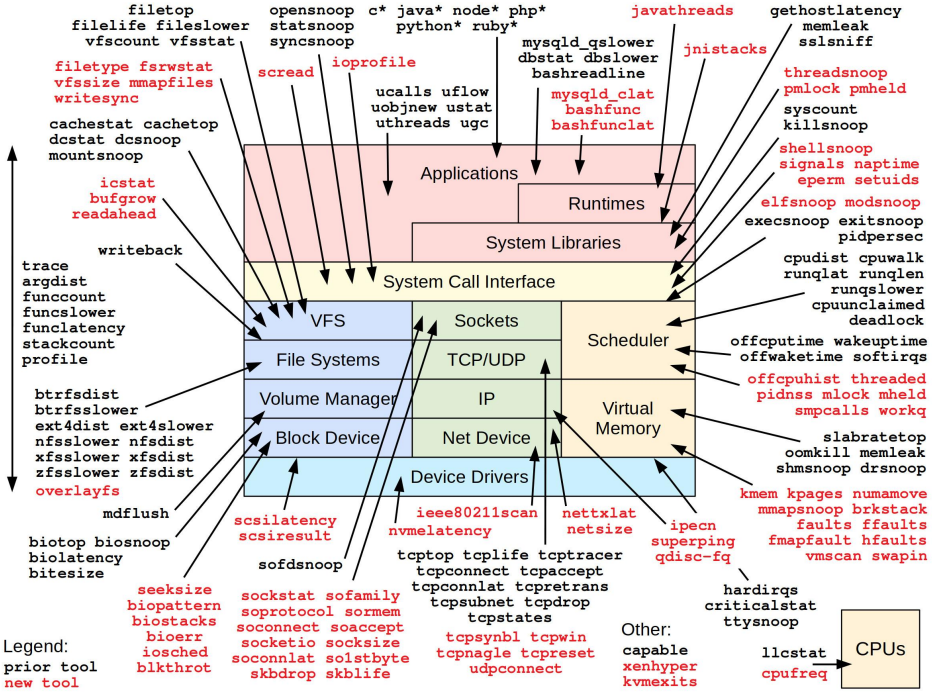
Writing eBPF in Go

Implementations for 

	Type	Base	Helper Functions
golang.org/x/net/bpf	classic BPF	Go	⛔
github.com/iovisor/gobpf github.com/dropbox/goebpf github.com/aquasecurity/libbpfgo	eBPF	C and Go	✅
github.com/cilium/ebpf	eBPF	Go	✅

https://github.com/iovisor/bpftrace

New tools developed for the book **BPF Performance Tools: Linux System and Application Observability** by Brendan Gregg (Addison Wesley, 2019), which also covers **prior BPF tools**



http://www.brendangregg.com/linuxperf.html

Tracing Go with eBPF

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

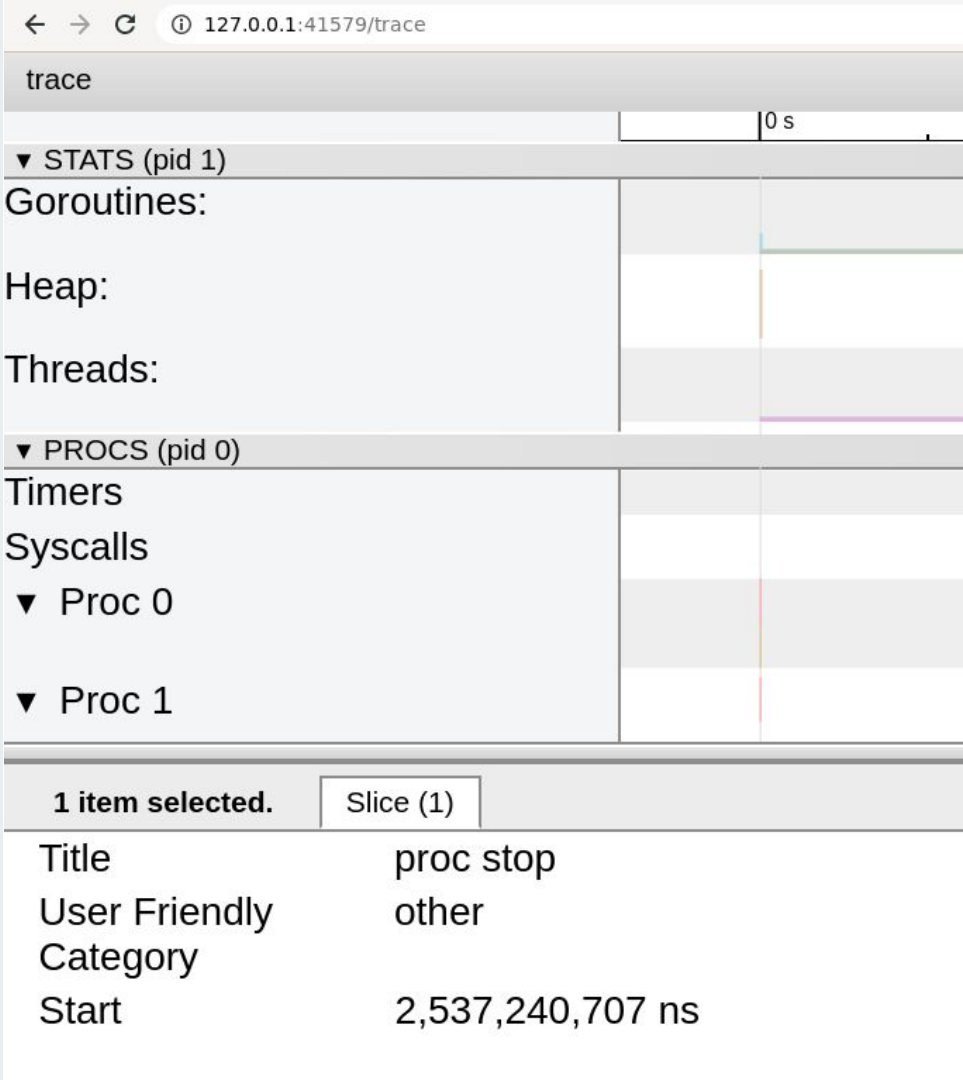
https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5



```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -l 'uprobe:/tmp/fibonacci:*'
uprobe:/tmp/fibonacci:main.fibonacciCache
uprobe:/tmp/fibonacci:main.fibonacciLoop
uprobe:/tmp/fibonacci:main.fibonacciRecursive
uprobe:/tmp/fibonacci:main.init.0
uprobe:/tmp/fibonacci:main.main
...
uprobe:/tmp/fibonacci:runtime.mallocgc
uprobe:/tmp/fibonacci:runtime.stackfree
...
uprobe:/tmp/fibonacci:sync.runtime_Semrelease
uprobe:/tmp/fibonacci:sync.runtime_canSpin
uprobe:/tmp/fibonacci:sync.runtime_doSpin
...
```

https://github.com/iovisor/bpftrace

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciLoop
{
  printf("arg: %d\n", sarg0);
}'

arg: 9
arg: 12
arg: 7
arg: 29
arg: 12
arg: 18
arg: 15
...
```

https://github.com/iovisor/bpftrace

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e 'uprobe:/tmp/fibonacci:main.fib*
{    printf("%s\n", ustack(perf)) }'
Attaching 3 probes...
```

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e 'uprobe:/tmp/fibonacci:main.fib*
{    printf("%s\n", ustack(perf)) }'
Attaching 3 probes...

    464260 main.fibonacciRecursive+0 (/tmp/fibonacci)
    4310d6 runtime.main+598 (/tmp/fibonacci)
    45cd41 runtime.goexit+1 (/tmp/fibonacci)

    464260 main.fibonacciRecursive+0 (/tmp/fibonacci)
    4644fb main.main+59 (/tmp/fibonacci)
    4310d6 runtime.main+598 (/tmp/fibonacci)
    45cd41 runtime.goexit+1 (/tmp/fibonacci)

    464260 main.fibonacciRecursive+0 (/tmp/fibonacci)
    4644fb main.main+59 (/tmp/fibonacci)
    4310d6 runtime.main+598 (/tmp/fibonacci)
    45cd41 runtime.goexit+1 (/tmp/fibonacci)

    4642e0 main.fibonacciCache+0 (/tmp/fibonacci)
    4310d6 runtime.main+598 (/tmp/fibonacci)
    45cd41 runtime.goexit+1 (/tmp/fibonacci)

    464400 main.fibonacciLoop+0 (/tmp/fibonacci)
    4310d6 runtime.main+598 (/tmp/fibonacci)
    45cd41 runtime.goexit+1 (/tmp/fibonacci)
...
```

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciRecursive
{
  @start[pid] = nsecs;
}
uretprobe:/tmp/fibonacci:main.fibonacciRecursive
/@start[pid]/
{
  @ns[comm] = hist(nsecs - @start[pid]);
  delete(@start[pid]);
}'
@ns[fibonacci]:
[2K, 4K)          973 |@@@@@@@@@@@@@@@@@@@@@@@@@@|
[4K, 8K)          326 |@@@@@@@@                    |
[8K, 16K)          27 |@                           |
[16K, 32K)          2 |                            |
[32K, 64K)          1 |                            |
[64K, 128K)         1 |                            |
```

https://github.com/iovisor/bpftrace

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciLoop
{
  @start[pid] = nsecs;
}
uretprobe:/tmp/fibonacci:main.fibonacciLoop
/@start[pid]/
{
  @ns[comm] = hist(nsecs - @start[pid]);
  delete(@start[pid]);
}'
@ns[fibonacci]:
[8K, 16K)          2 |@@@@@@@@@@@@@@@@@@@@|
[16K, 32K)         1 |@@@@@@@@@                |
```

https://github.com/iovisor/bpftrace

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace  -e 'uprobe:/tmp/fibonacci:runtime.sig*
{
	printf("%s()\t%d%s\n", func, sarg0, ustack(perf))
}'
Attaching 20 probes...
```

https://github.com/iovisor/bpftrace

```go
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace  -e 'uprobe:/tmp/fibonacci:runtime.sig*
{
	printf("%s()\t%d%s\n", func, sarg0, ustack(perf))
}'
Attaching 20 probes...

runtime.sighandler()	10

	442580 runtime.sighandler+0 (/tmp/fibonacci)
	45e8a3 runtime.sigtramp+67 (/tmp/fibonacci)
	45e9a0 runtime.sigreturn+0 (/tmp/fibonacci)
	43630e runtime.findrunnable+1006 (/tmp/fibonacci)
	437997 runtime.schedule+727 (/tmp/fibonacci)
	437f1d runtime.park_m+157 (/tmp/fibonacci)
	45b03b runtime.mcall+91 (/tmp/fibonacci)
	45a072 time.Sleep+210 (/tmp/fibonacci)
	464653 main.main+403 (/tmp/fibonacci)
	4310d6 runtime.main+598 (/tmp/fibonacci)
```

SIGUSR1

https://github.com/iovisor/bpftrace

Questions?

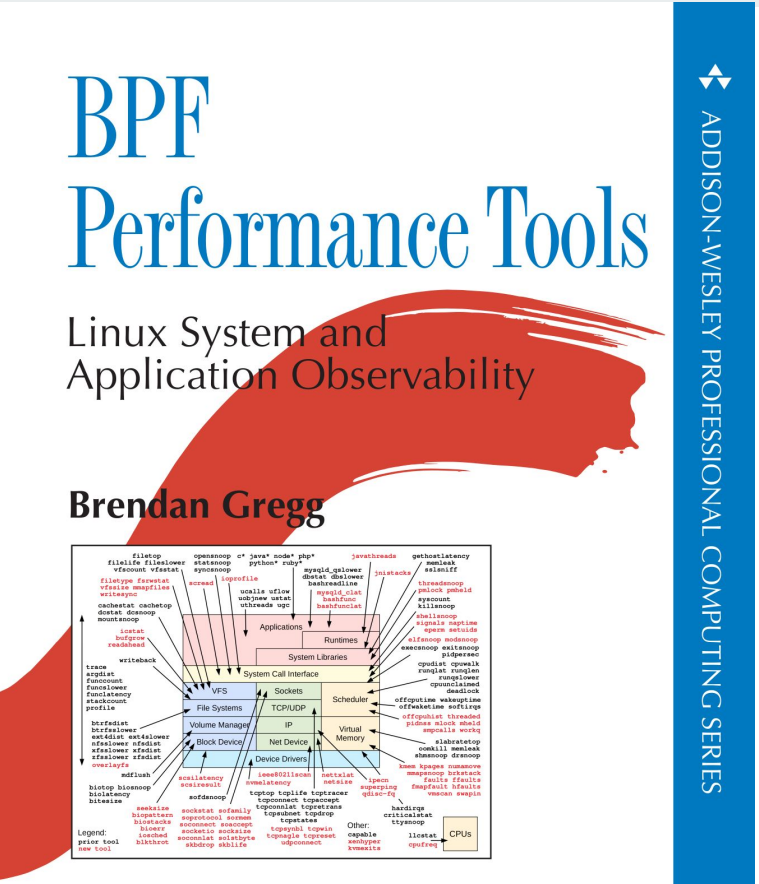
Twitter
https://twitter.com/0x0F10

Github
https://github.com/florianl

Slides
https://github.com/florianl/talks

Gophers by
github.com/ashleymcnamara/gophers


https://www.pexels.com/photo/blur-book-stack-books-bookshelves-590493/



Resources

- https://ebpf.io/
- BPF Performance Tools (Book)
- BPF and XDP Reference Guide